# USER MANUAL OF VCD 2.0 PACKAGE

## Abstract

Detailed data processing and network training & validation of VCD 2.0

Written by Yi Chengqiang at FeiLab

cqyi@hust.edu.cn

# Contents

# Figures and Tables

1. **What is VCD 2.0 package?**

   VCD 2.0 package is designed for 3D super-resolution reconstruction in light-field microscopy. It's based on the physics-informed multi-stage neural network that can reconstruct the signal distribution from severely degraded 2D light-field captures. To be user-friendly, a GUI (graphic user interface) is contained in this package to achieve data-processing & neural network construction step-by-step.

2. **Running environment installation**

   The source codes of VCD 2.0 were written in MATLAB and Python. So, the installation of running environment of this package contains 2 steps:
   a. Download MATLAB from http://www.mathworks.com/products/matlab and install.
      **Critical:** Recommended version of MATLAB was 2022b (or later version). Besides, Parallel Computing Toolbox and Image Processing Toolbox must be chosen when installing MATLAB.
   b. DL-model required running environment installation.

      We recommend to use Anaconda to manage different running environments. Users can download this software at https://docs.conda.io/en/latest/.

      i. To install the required dependencies, we provide two ways to fast install corresponding packages:

      **_Option A_**: Download the compressed file of virtual environment from Google drive. Users can directly unzip this file into their own environment folder (e.g. '*./anaconda3/envs/*'). Once installed this packaged environment, users need not to follow the installation step in "*Option B*".

      _Option B_: Use '. *yml*' file that includes all packages' name and version to create virtual environment. Users need to type one command in Anaconda Prompt：
          > conda env create -f environment.yml

      **Critical**: VCD 2.0 network is written by Tensorflow 1.x. If using RTX30 or RXT40 series GPU, users need to install specific version of Tensorflow-GPU for Ampere GPU, which can be downloaded from tensorflow-windows-wheel.

      ii. Install *Cuda* and *cuddn.* We test these codes on Cuda v11.1 and cuddn v8.2.0.

3. **Workflow of VCD 2.0 package**

3.1  Overview of VCD 2.0 package

   This package is mainly composed of two components: GUI code written in MATLAB and DL network code written in Python as Fig 1 shows. Users can run "*main.mlapp*" in MATLAB to acess "data generation", "network training", "network validation" and "fine-tuning".
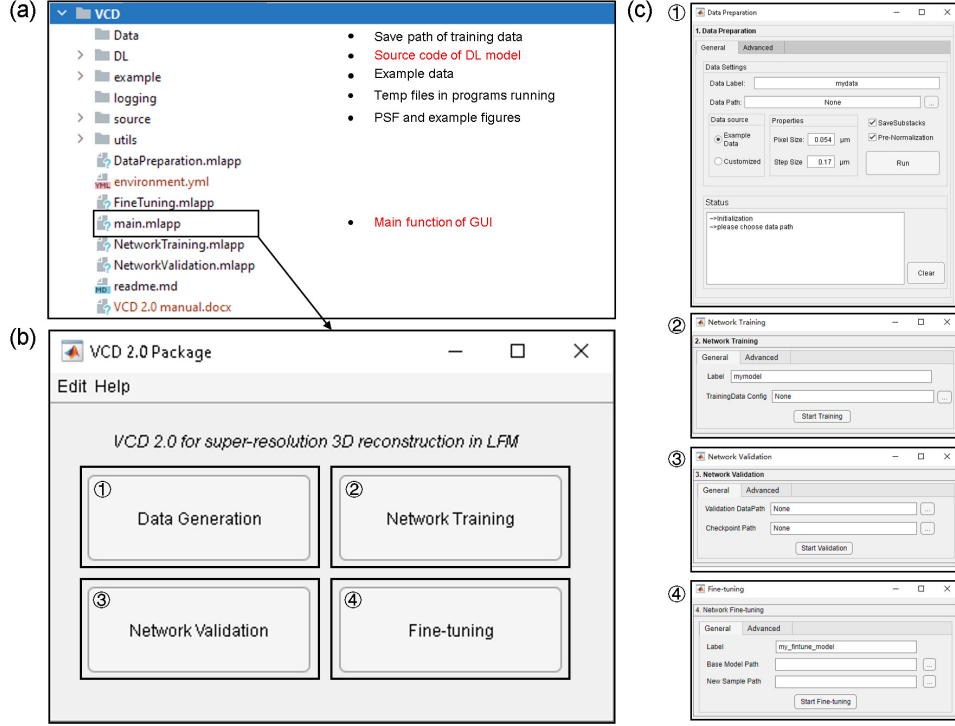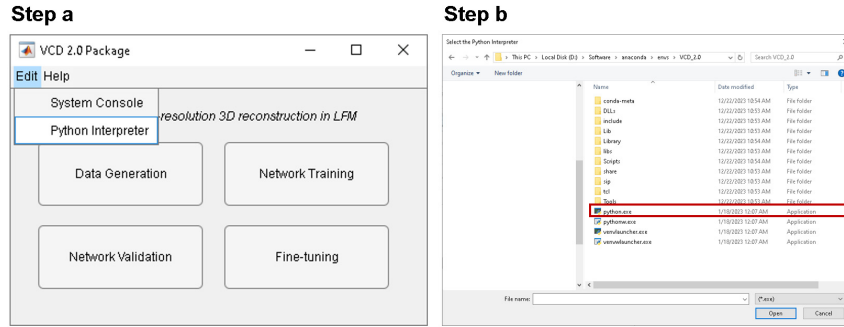
Fig. 1 Overview of VCD 2.0 package. (a) Directory structure (b) GUI schematic of VCD 2.0 (c) GUIs of four modules

Before VCD 2.0 implementation, users need to load the installed environment in GUI panel:

a. Click "Python Interpreter" from "Edit->Python Interpreter"
b. Choose the "python.exe" in the folder of installed environment.



**Note**: The path of loaded interpreter will be saved at "*./logging/python_interp_dir.mat*". Users can delete this file and reload the interpreter if the loaded environment is not desirable.

Next, we will introduce the detailed operations in each module.

## 3.2 Data preparation

This module is used to generate training dataset, which includes "Noisy LF", "Clean LF", "De-aliased LF" and "super-resolution 3D stacks", referred to the physical degradation process in light-field microscopy imaging. The screenshot of this module is shown in Fig. 2. The detailed descriptions of parameters in this module are listed in Table 1.
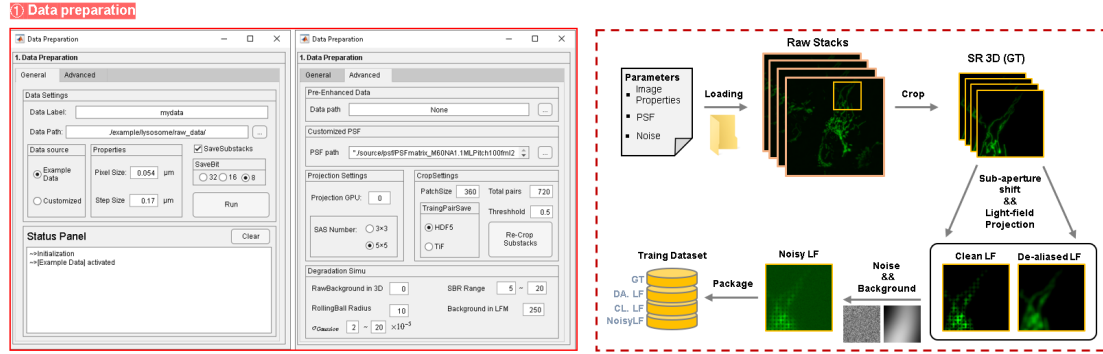
Fig. 2 Overview of Module "Data preparation"

| General | |
|---|---|
| Data label | The folder name of saved data |
| Data path | The patho of input raw 3D data |
| Data source | The type of input data. Example data: AiryScan data. If 'customized' is enable, users need to load LF PSF in Advanced Panel. |
| pixel size | The pixel size of input data |
| step size | The z-step size of input data |
| SaveSubstacks | Whether to save sub-stacks or not. (Ture, recommended) |
| SaveBit | The bitdepth of saved dataset. Note: "32 bit" denotes the pre-normalization (0~1.0) on cropped data. |
| Advanced | |
| Data path (Pre-enhanced Data) | The path of enhanced data. In VCD 2.0, we used IsoNet to enhanced the axial resolution of raw SR data. |
| PSF path | The path of customized PSF. Default: example PSF. |
| Projection GPU | The id number of GPU. |
| SAS Number | The times of sub-aperture shifts. |
| PatchSize | The patch size of cropped LF |
| Threshold | The threshold in data cropping process to decide whether to save cropped data or not |
| Total pairs | The total number of cropped pairs |
| TrainingPairSave | The file type of saved cropped data. (HDF5: packaged and pre-normalized data; TIF: image data) |
| RawBackground in 3D | The constant background value of input 3D stacks. |
| SBR Range | The range of signal-to-background ratio in genreated Noisy LF, which should be coincided with the SBR of caputred LFM |
| RollingBall radius | The radius of rolling ball subtraction algorithm for reducing the background in raw 3D stacks. Note: If enhanced data allowed, this operation will be disabled |
| Background in LFM | The constant background value for degraded light-field images simulation, which should be the same with experimental one. |
| Gaussion Noise (σ) | The standard deviation of gaussion noise. |

Table 1 Descriptions of parameters in Module "Data Preparation"

Example Usage:

Specifically, we will detail an example step-by-step to introduce the whole process of data generation. Users can download the example training data in Google drive, and extract it to the folder "./*example*".

    a. Edit the label and choose the path of raw data via the button [...]

    b. Choose the data type of input data.

Here, we used the Airyscan data as an example. The program will automatically set the properties and corresponding default PSF matrix of light-field microscopy. The default matrix is saved at "./*source*/*psf*".
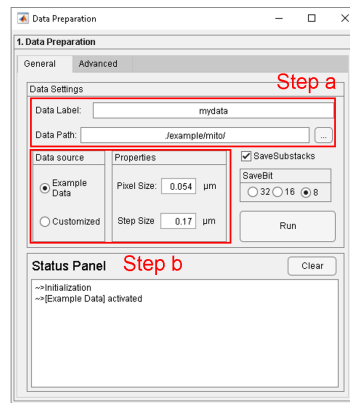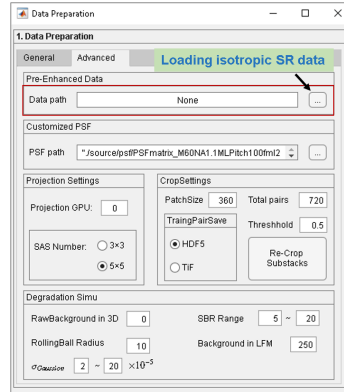


**Note:** Users can choose the "customized" option and edit the properties of input data. Then,

choose the PSF path in "Advanced panel".

c.  *(Optional)* Loading the enhanced SR 3D data.

   Due to the axial resolution limitation of Airyscan microscopy, we used an IsoNet to obtain an enhanced SR data with isotropic resolution. Users can load the enhanced data in Advanced panel.



d.  Set the degradation parameters in "Advanced panel"

   For our provided example data, we used the following settings:

   · *RawBackground in 3D*: 0 (There existed less constant background offset in airyscan data)

   · *RollingBall Radius:* 10 (Remove the auto-fluoresce and defocus signal)

   · *Background in LFM:* 250

   · *SBR Range:* 1~10 (Random SBR makes the model more robust)

   · *Gaussian Noise sigma:* 2~20 $\times 10^{-5}$

e.  Finish the "*Projection Settings*" and "*CropSettings*"

   In this example, we adopted the following settings:

   · *SAS Number*: 25 (The angular sampling of our setup is 15 which produces the extremely under-sampled views. We used $5\times5$ sub-aperture shifts strategy to generate high-resolution LF.)

   · PatchSize: 360 (decided by the maximum memory of GPU. Larger size will be better.)

   · Threshold: 0.5

   · Total training pairs: 720 (Here, ~20 cells were used to generate training dataset. We recommend to set this value as 700~1500. Feeding more samples produces better results)

   **Note:** Users need to check their own data to derive above parameters. For example, the "*Background in LFM*" and "Gaussian noise sigma" can be derived from the measurement of the mean value and standard deviation of background area in experimental data. "*RollingBall Radius*" can be decided by processing the raw stacks with "Subtract background" in Fiji.

f.  Click "*Run*" and wait.

   The whole process commonly lasts several hours. During data generation, the status of program running will be shown in "Status Panel", for example, the projection operation and cropping operation.
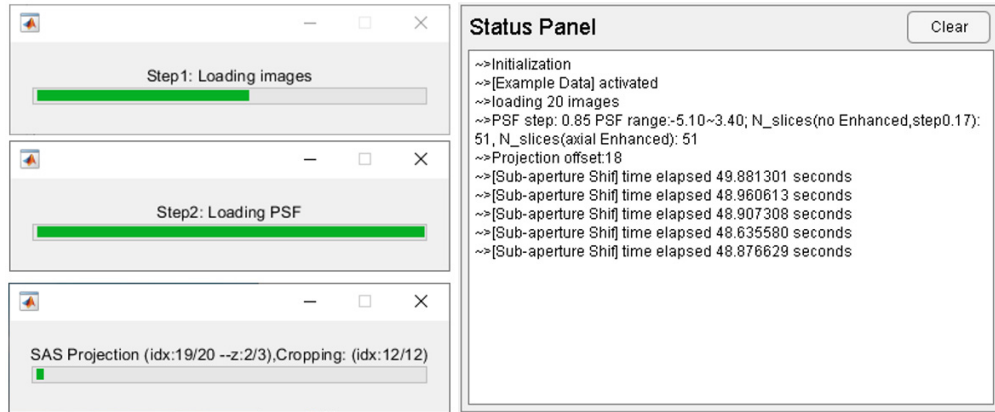
Fig. Program running monitoring

When the data preparation was finished, users can check the generated images in corresponding folder, e.g., "./*Data*/{*your label name*}/*S01_TrainingData*" or "./*Data*/{*your label*}/ *S00_Multi_stage_data*" (If "*SaveSubstacks*" is enable.). Besides, the detailed parameters in this module are also saved (seen at "./*Data*/{*your label*}/*Data_details.json*").
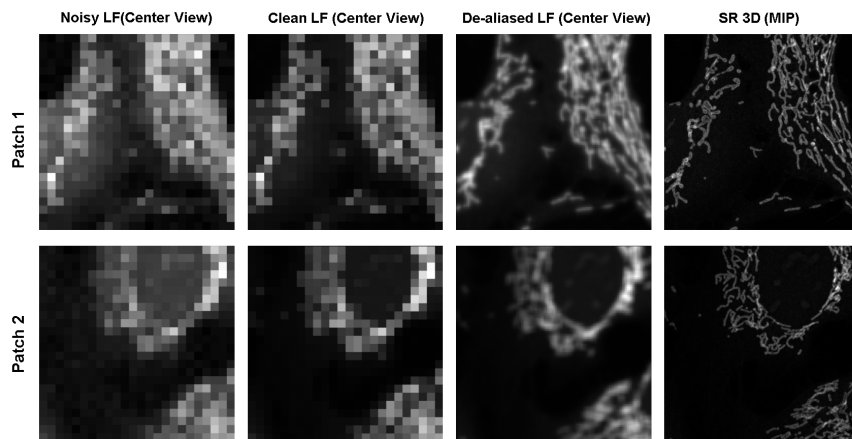


Fig. Sample data from generated training dataset



Fig. Generated files when "data preparation" finished

## 3.3  Network Training

This module is used train an VCD 2.0 model with the supervision of generated data in last module. Click the "Network Training" in "main" GUI, this module will pop up as Fig. 2 shows. The detailed explanations of the parameters in this module are listed in Table 2.
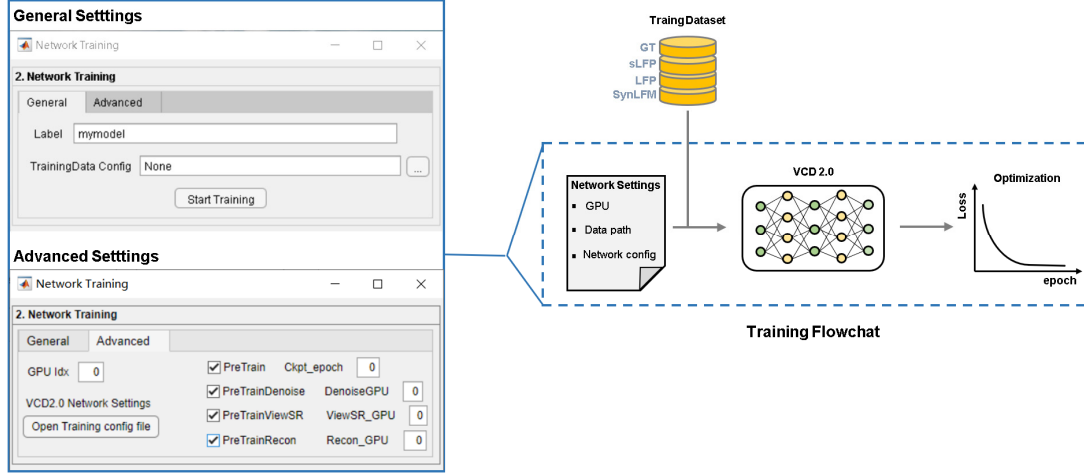
Fig. 3 Overview of Module "Network Training"

| General | |
|---|---|
| label | The folder name for model saving |
| TrainingData Config | The config files of training data, e.g. "./Data/mydata/data_details.json" |
| Advanced | |
| GPU Idx | The id number of GPU for joint-optimization of VCD 2.0 |
| PreTrain | Whether to pretrain network modules in VCD 2.0 |
| PreTrainDenoise | Whether to pretrain LF Denoising network |
| PreTrainViewSR | Whether to pretrain LF De-aliasing network |
| PreTrainRecon | Whether to pretrain 3D reconstruction network |
| DenoiseGPU | The id number of GPU for LF Denoising network pretraining |
| ViewSR_GPU | The id number of GPU for LF De-aliasing network pretraining |
| Recon_GPU | The id number of GPU for 3D Reconstruction network pretraining |

Table. 2 Explanations of parameters in Module "Network Training"

Example Usage:

Based on the generated training dataset in last example usage, we would show how to train an VCD 2.0 model:

a. Edit the label and choose the path of training data via the button  ...

b. Assign the GPUs to networks training at "Advanced" panel

**Note:** Due to the high difficulty of decoding SR signals from degraded LF, the DNN model must be deeper enough to construct function sets to fit the feeding data, which poses a challenge on network optimization in training process. Here, we adopt pre-training strategy commonly seen in DL-based vision processing to avoid undesirable opmization process, e.g. local optimum, premature convergence and even divergence. Users can select the "Pretrain" checkbox to enable pre-train each module in VCD 2.0 as the following figure shows. Besides, users can designate specific GPU to the training task of some module. We recommend to use different GPUs to achieve parallel pre-training.
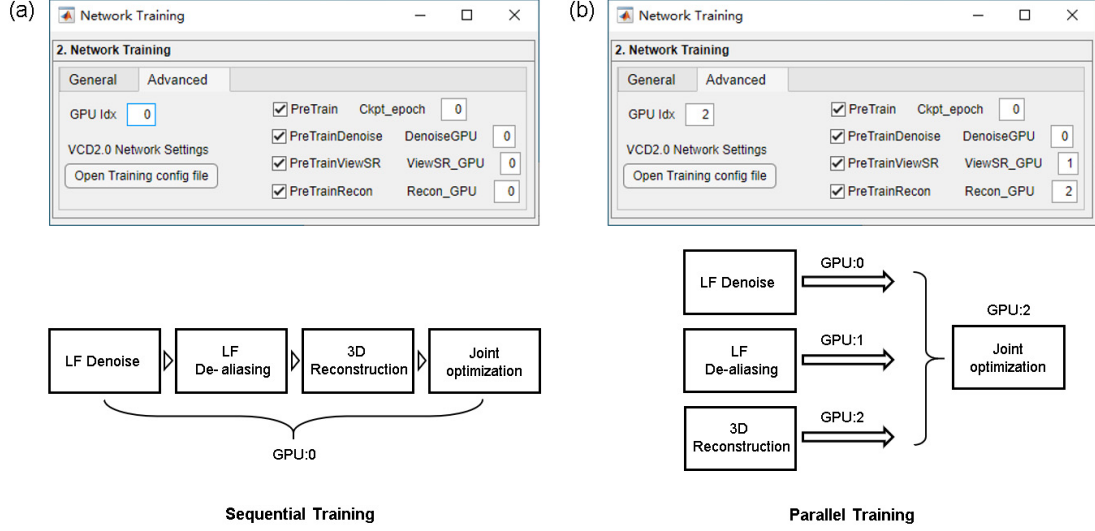
Fig. Pre-training settings

c.   (Optional) Customize the hyper parameters of VCD 2.0, e.g. learning rate, training epochs, network structure, etc. Users can click "Open Training config file" to modify the hyper-parameters in Joint-optimization as the following screenshot shows. For pre-training settings, users need to open each ".*py*" file (e.g. "*preTrain_denoise.py*") and modify the default parameters in "main" function.
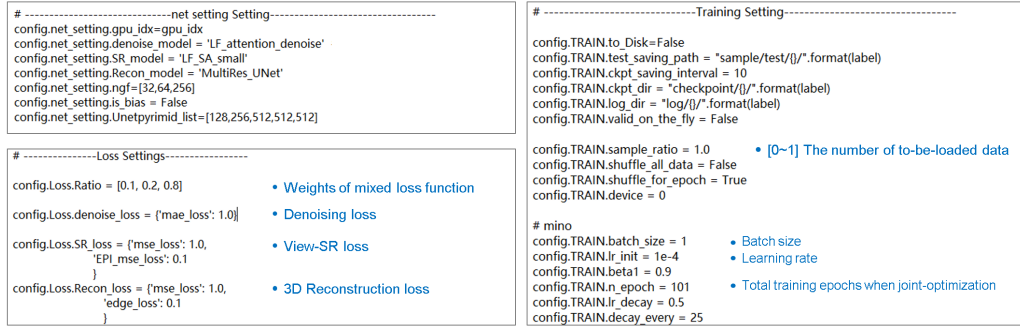


Fig. Hyper parameters in Joint-optimization

d.   Click "Start Training" and the consoles will pop up to display the training process of each training tasks.

**Note**: During training, users can check the intermediate results in the folder "*./DL/sample/test/{your label}*". Besides, corresponding checkpoints and training logs are also saved in the folder "*./DL/ checkpoints*" and "*./DL/logs*", respectively.

## 3.4  Network Validation

This module is used to reconstruct 3D signals from input LFs via trained VCD 2.0 network. The screenshot of "Network Validation" module is seen in Fig. 4. The parameters in this module are detailed in Table 3.
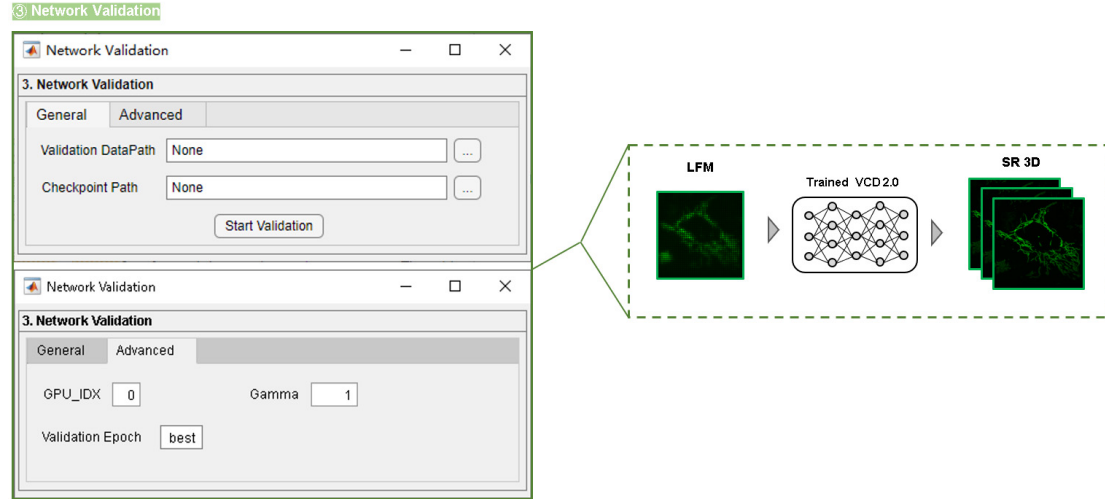
Fig. 4 Overview of Module "Network Validation"

| General | |
|---|---|
| Validation DataPath | The folder name of input LFs |
| Checkpoint Path | The path where trained model saved in. |
| Advanced | |
| GPU Idx | The id number of GPU for validation of VCD 2.0 |
| gamma | Apply the grayscale mapping with the gamma curve. |
| Validation Epoch | The epoch of saved checkpoints for validation. "best" means the loaded checkpoint achieved lowest validation errors. Users can also enter specific number to load corresponding checkpoint, e.g. 100. |

Table. 3 Descriptions of parameters in Module "Network Validation"

Example Usage:

**Note:** For quick validation of VCD 2.0, users can use our provided example data (located at './*example*/*validation_data*') and corresponding trained VCD 2.0 models (located at './*DL*/*checkpoint*') to evaluate its performance. Among these models, "*mito_enhanced*" denotes the model trained on outer membrane of mitochondria data; "*lysosome_enhanced*" denotes the model trained on outer membrane of lysosome data; "*mito2matrix_finetuning*" denotes the fine-tuned model with the aid of WF images. The reconstructed results can be downloaded from Google drive

a. Choose the path of validation data via the button […]

b. Choose the path of trained model via the button […], e.g. "./*DL*/*checkpoint*/{*model label*}".

c. (Optional) Edit the GPU id, validation epoch and gamma.

d. Click "Start Validation", and the reconstruction results will be automatically saved at the same folder of input LFs.

## 3.5  Fine-tuning

This module was designed to extend the utility of VCD 2.0 network in the absence of high-resolution 3D stacks as ground truth. Here, we adopted 2D wide-field captures as structural constrains to remove the hallucinations when meeting mismatched data. Fig.4 shows the overview of module "Fine-tuning"
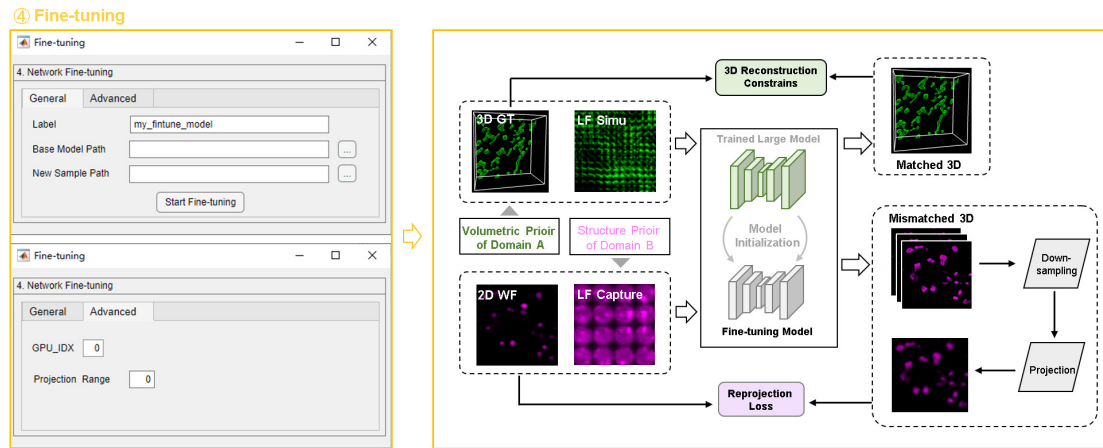
④ Fine-tuning



Fig. 5 Overview of Module "Fine-tuning"

| General | |
|---|---|
| label | The name of new model |
| Base Model Path | The path of trained model path |
| New Samle Path | The path of paired WF-LF images of new sample |
| Advanced | |
| GPU Idx | The id number of GPU for network fine-tuning |
| Projection Range | The axial range for generating re-projection in the process of reprojection loss computation. (Default: 0, means all slices were used to compute re-projections; Nonzero: 0~projection range) |

Table 4. Explanations of parameters in Module "Fine-tuning"

Example usage:

Here, we will show how to achieve transfer learning based on trained model (training data: mitochondria outer membrane) and reconstruct new data (mitochondria matrix) only under the supervision of 2D WF-LF.

**Note:** We recommend to pre-process the WF images, *e.g.* background subtraction, to reduce the influence from non-signal region.

**Critical:** The WF and LF patches must be saved in "./{folder name}/WF" and "./{folder name}/LF", respectively. Besides, the patch size must be the same with the size of training data of trained model. For example, previous settings shown the cropped size was set as 360, which required the size of input LF && WF of new sample was 360.

a.   Edit the label and choose the path of base model and new sample via the button [...]
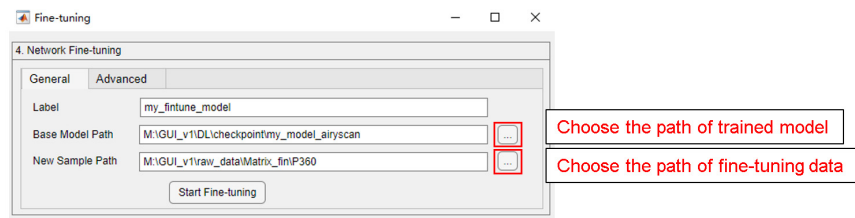


b.   (Optional) Edit the GPU number and projection range.

c.   Click "Start Fine-tuning", and the fine-tuning process will be automatically started. Users can also check the intermediate results in "./*DL*/*sample*/*test*/*{your label}*".